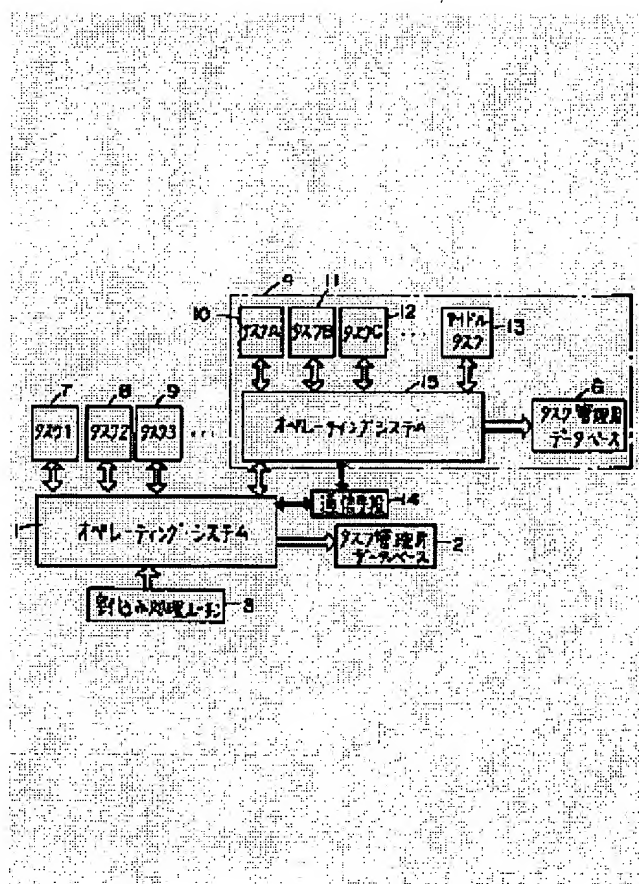


DATA PROCESSING SYSTEM

Patent number: JP5108380
Publication date: 1993-04-30
Inventor: SATO KOICHI
Applicant: MITSUBISHI ELECTRIC CORP
Classification:
- international: G06F9/46
- european:
Application number: JP19910272846 19911021
Priority number(s):

Abstract of JP5108380

PURPOSE: To provide the data processing system without deteriorating high-speed responsiveness even when the function of the system is more improved by providing independently first and second execution control data management means on the first and second execution control means.
CONSTITUTION: The peripheral system of an OS (operating system) 15 operates as an idle task 4 under the management of the OS 1. The execution is shifted to the OS 15 when tasks 7 to 9 under the management of the OS 1 are finished, and the tasks 10 to 13 managed by the OS 15 are executed. When tasks 10 to 13 issue a system call, the OS 15 executes the processing. When a system call is generated, the context of the task under execution is saved in a database 6 for task management. The task scheduling is performed by the OS 15, and the data of the database 6 is operated. The context of a newly scheduled task is restored in a register group of hardware for task execution from the database 6.



Data supplied from the esp@cenet database - Worldwide

(43)公開日 平成5年(1993)4月30日

F 8120-5B

審査請求 未請求 請求項の数 1 (全 9 頁)

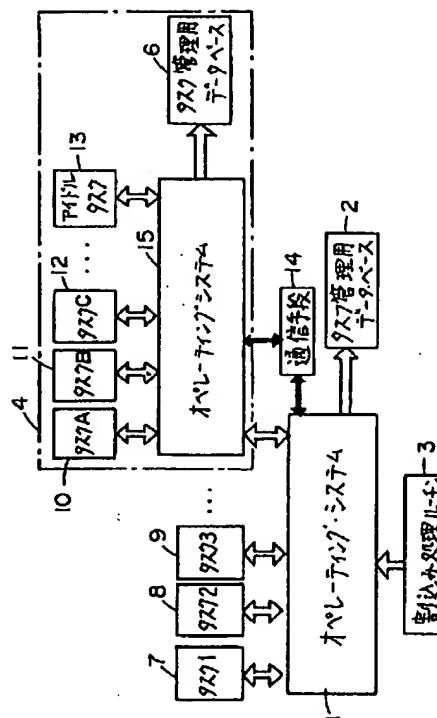
(74)代理人 弁理士 深見 久郎 (外3名)

(54)【発明の名称】 データ処理システム

(57) 【要約】

【目的】 システムの機能がより高度化しても、高速応答性が低下しないデータ処理システムを提供する。

【構成】 タスク 4、7～9を制御する OS 1 と、OS 1 のタスク管理用のデータベース 2 と、OS 1 にタスクの 1 つとして割付けられ、タスク 10～13 の実行制御を行なうための OS 5 と、OS 5 のタスク管理のためのデータベース 6 とを含む。タスクを、そのコンテキストの大きさに応じて小さなものは OS 1 に、大きなものは OS 5 に割振ることが可能となる。



【特許請求の範囲】

【請求項 1】 データ処理に必要なハードウェア資源およびソフトウェア資源と、

複数個の第 1 の仕事単位の各々に、所定の基準に従って前記ハードウェア資源およびソフトウェア資源を割当てることにより、各前記第 1 の仕事単位の実行状態を制御するための第 1 の実行制御手段と、

前記第 1 の実行制御手段による前記第 1 の仕事単位の実行状態の制御に必要なデータを格納して管理するための第 1 の実行制御データ管理手段と、

前記第 1 の実行制御手段に、前記第 1 の仕事単位の 1 つとして割付けられ、複数個の第 2 の仕事単位の各々に、所定の基準に従って前記ハードウェア資源およびソフトウェア資源を割当てることにより各前記第 2 の仕事単位の実行状態を制御するための第 2 の実行制御手段と、

前記第 2 の実行制御手段による前記第 2 の仕事単位の実行状態の制御に必要なデータを格納して管理するための第 2 の実行制御データ管理手段とを含むデータ処理システム。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 この発明は主としてコンピュータなどからなるデータ処理システムに関し、特に、高速応答性が要求されるリアルタイム処理のためのデータ処理システムに関する。

【0002】

【従来の技術】 従来のデータ処理システムの多くは、リアルタイム処理機能を備えたオペレーティングシステム（以後「OS」と省略する）により動作している。OS は、コンピュータシステムの運用効率と操作性を高めるために、コンピュータシステムに組込まれたシステムプログラムを主として意味する。

【0003】 図 6 は、従来のデータ処理システムのソフトウェア構成の概念図である。図 6 を参照して、このデータ処理システムはソフトウェアとして、リアルタイム OS 60 と、OS 60 の制御下で動作するタスク 63～66 と、これらタスク群 63～66 の実行を管理するために OS 60 が使用するタスク管理用データベース 61 と、オペレーティングシステム 60 に対する割込の発生によって実行される割込処理ルーチン 62 とを含む。

【0004】 各タスク 63～66 は、互いに独立した応用プログラムの最小仕事単位であり、必要に応じて個々に生成される。OS 60 はこれらのタスク 63～66 の実行制御を管理し、いわゆるマルチタスキング処理を行なう。各タスク 63～66 には予め優先順位が与えられている。リアルタイム OS 60 は、各タスクに与えられた優先度に応じて、実行可能なタスクのうち最も高い優先度を有するタスクに実行権を与える。

【0005】 たとえば、現在実行中のタスクが OS の提供する機能を利用しようとする場合を考える。この場合

実行中のタスクは、システムコールと呼ばれる、システムに対する要求を発行する。実行中のタスクはシステムコールを発行した後、一旦待ち状態となる。実行中のタスクが待ち状態となれば、CPU が空き状態となる。資源を有効に利用するためには、この間に CPU を他のタスクに割当ててこのタスクを実行させればよい。このとき、実行可能な複数のタスクがある場合、最も優先度の高いタスクが選択され実行される。また、現在実行中のタスクより優先度の高いタスクが実行可能状態になったとき（待ち状態が解除されたとき）は、現在実行中のタスクが一旦待ち状態にされ、その優先度の高いタスクに実行が移される。

【0006】 各タスクの状態はシステムコールにより遷移する。システムコールは実行中のタスクまたは割込処理ルーチンにより発行される。図 7 はその一例を示す状態遷移図である。図 7 を参照して、タスク 1 は優先度の低いタスク、タスク 2 は優先度の高いタスクであるものとする。時刻 t_1 以前にはタスク 1 が実行状態であり、タスク 2 が待ち状態になっているものとする。時刻 t_1 でタスク 1 がタスク 2 の待ち状態を解除するためのシステムコールを発行したとする。するとこの時点で実行は一時 OS に移る。OS はタスク 1 からのシステムコールを受けてタスク 2 を実行可能状態とする。さらに OS は、タスク 1 とタスク 2 との優先度を比較して、次にどちらのタスクを実行すべきか判断する。どのタスクを実行すべきか判断することをタスクのスケジューリングと呼ぶ。その結果、優先度の高いタスクが時刻 t_2 に実行開始される。

【0007】 OS がタスクのスケジューリングを行なう場合、タスクの実行状態や優先度などを表わす情報を記憶する必要がある。そのために OS は、図 6 に示されるタスク管理用データベース 61 を使用する。さらに OS は、複数のタスク間の通信や同期を実現するために、フラグやセマフォと呼ばれる様々なマルチタスキングのための機能を提供する。これらの機能を実現するためには、システムに共通のデータベースが必要とされる。このデータベースは、タスク管理用データベース 61 に含まれる。OS が各タスクに対して提供する機能が複雑になるほどタスク管理用データベース 61 に格納すべき情報は増加する。また、それにつれて 1 回のシステムコールの発生によって操作されるデータ量も増加する。

【0008】 図 7 において、OS による処理の前後でタスク 1 からタスク 2 に実行が移っている。このことを「タスクをディスパッチする」と呼ぶ。タスクのディスパッチは、OS により以下に行なわれる。まず、ディスパッチ前のタスクが使用していたハードウェアのレジスタの内容を図 6 のタスク管理用データベース 61 にセーブする。次に、ディスパッチ後に実行されるタスクのレジスタ内容を、タスク管理用データベース 61 からタスクを実行するためのハードウェアのレジスタにリ

ストアする。タスク管理用データベース61には、このディスパッチ後に実行されるタスクが前回処理中断したときのハードウェアのレジスタ内容が格納されており、リストアにより前回の処理中断時の状態にハードウェアの各レジスタ内容が戻る。セーブ・リストアされるハードウェアのレジスタとしては、MPU（マイクロプロセッサユニット）の制御レジスタや汎用レジスタ、FPU（浮動小数点演算ユニット）の浮動小数点レジスタなどがある。これらハードウェアのレジスタ内容をタスクのコンテキストと呼ぶ。

【0009】システムコールは実行中のタスクのほか、に、割込処理ルーチンからも発行される。この場合もシステムコールが発行されると一時実行はOSに移る。OSはこのシステムコールに応答して対応する処理を行なう。

【0010】ところで、OSがシステムコールを処理しているときに割込を無制限に受け付けると、割込処理の中でタスク管理用データベース61の内容が操作され、データベースの一意性が失われる可能性がある。このような事態となればタスク管理は不可能となる。これを防ぐために、通常、OSが上述のシステムコールに対する処理を行なっているような場合には、割込は禁止される。ただし、リアルタイムOSは高速応答性を備えていることが絶対条件である。特に割込禁止時間が長くなると、外部要因に対するシステムの応答性能に直接悪影響を及ぼし好ましくない。そのため、リアルタイムOSの設計時には、割込禁止時間が可能な限り短くなるよう最大の注意が払われる。

【0011】図8は、システムコールが発行されたときのOSの処理の一例を示すフローチャートである。システムコール入口からこの処理に入った場合、ステップS81でまず割込が禁止される。続いてステップS82で、実行していたタスクのコンテキストをデータベース（タスク管理用データベース61）にセーブする。さらにステップS83で、データベース61を操作してタスクのスケジューリングが行なわれる。ステップS84で、スケジューリングの結果次に実行されるタスクとして選択されたタスクのコンテキストをタスク管理用データベース61からタスク実行用のハードウェアのレジスタ群にリストアする。さらにステップS85で割込禁止が解除される。そしてシステムコールの処理は終了する。

【0012】図6において、OSにより実行制御されるタスク66は、「アイドルタスク」と呼ばれるものである。アイドルタスク66は、システムにおいて最も優先度の低いタスクである。このアイドルタスクを常にシステム内に存在させておくことにより、このシステム内において実行されるべきタスクが1つもないという状態があり得なくなる。

【0013】

【発明が解決しようとする課題】従来のデータ処理システムで、システムの高機能化に伴いOSが複雑な機能を提供する必要が増大している。この場合OSが操作するデータベースは大きくなり、その結果一度のシステムコールを処理するための時間は長くなり、コンテキストのセーブ・リストア時にオーバーヘッドが生ずる。マルチタスキング処理においてタスクの実行以外に必要な時間が増大することとなり、高速応答性を備えるべきリアルタイムOSとしては好ましくない。また、割込禁止時間も長くなり、割込に対する時間的な応答性が劣化する。

【0014】それゆえにこの発明の目的は、システムの機能がより高度化しても、高速応答性が低下しないデータ処理システムを提供することである。

【0015】

【課題を解決するための手段】本発明に係るデータ処理システムは、データ処理に必要なハードウェア資源およびソフトウェア資源と、複数の第1の仕事単位の各々に、所定の基準に従ってハードウェア資源およびソフトウェア資源を割当てることにより、各第1の仕事単位の実行状態を制御するための第1の実行制御手段と、第1の実行制御手段による第1の仕事単位の実行状態の制御に必要なデータを格納して管理するための第1の実行制御データ管理手段と、第1の実行制御手段に、第1の仕事単位の1つとして割付けられ、複数の第2の仕事単位の各々に、所定の基準に従ってハードウェア資源およびソフトウェア資源を割当てることにより、各第2の仕事単位の実行状態を制御するための第2の実行制御手段と、第2の実行制御手段による第2の仕事単位の実行状態の制御に必要なデータを格納して管理するための第2の実行制御データ管理手段とを含む。

【0016】

【作用】上述のデータ処理システムにおいては、第1の仕事単位と第2の仕事単位とは、それぞれ別個の第1の実行制御手段および第2の実行制御手段によって実行制御される。各実行制御手段の提供する機能を異ならせ、あまり多くの機能を要求しないタスクを実行制御するための実行制御手段には比較的少ない機能を、より多数の機能を提供する必要がある実行制御手段には比較的多数の機能をそれぞれ備えるように各実行制御手段の内容を異ならせることができる。また、第1の実行制御手段および第2の実行制御手段の各々に別個の実行制御データ管理手段を設けたため、各実行制御手段におけるタスク管理の際のコンテキストの内容を異ならせることができる。したがって、特定のタスクを、そのタスクがシステムに対して要求する機能やタスク管理のために必要なコンテキストの内容に応じて最も適切な実行制御手段の管理の下で実行させることができる。

【0017】

【実施例】図2は、この発明の一実施例に係るデータ処理システムのハードウェア構成図である。図2を参照し

て、このデータ処理システム27は、プログラムを実行するためのMPU20と、MPU20に接続され、データ処理システム27に含まれる各構成要素間でデータをやりとりしたり、MPU20に対して割込信号を入力したりするための内部バス26と、内部バス26に接続され、浮動小数点演算を行なうためのFPU28と、本システムを制御したり、データ処理を行なったりするためのプログラムおよびシステムを管理するためのデータベースと、処理すべきデータなどを格納するためのメモリ21と、内部バス26に接続され、CRT29を制御するためのCRTコントローラ22と、内部バス26および外部のネットワーク30に接続され、通信制御を行なうための通信コントローラ23と、内部バス26に接続され、ディスクユニット31などの外部記憶装置の制御を行なうためのディスクコントローラ24と、内部バス26に接続され、各種センサ32からの信号を変換してMPU20に与えるためのセンサコントローラ25とを含む。

【0018】図2に示される各構成要素のうち、各コントローラ22～25は、たとえばデータの入出力が完了したり、外部で変化が生じたことを検知したときに、割込信号を内部バス26を介してMPU20に与える。また、データ処理に必要なソフトウェア資源およびタスク管理のためのオペレーティングシステムなどのシステム資源もメモリ21に格納されている。

【0019】図1は、この発明の一実施例に係るデータ処理システムを形成するソフトウェア資源の構成図である。このソフトウェア資源は、前述のように図2に示されるメモリ21に格納されている。図1を参照して、このシステムは、制限された機能のみを各タスクに対して提供するリアルタイムOS1と、OS1の管理下で動作するタスク7～9と同じくOS1の管理下で動作するアイドルタスク4と、OS1がタスク4、7～9の実行制御を管理するために用いるためのタスク管理用データベース2と、OS1の管理下で処理される割込処理ルーチン3とを含む。

【0020】アイドルタスク4は1つのシステムを構成しており、OS1のアイドルタスクとして実行される、OS1よりも複雑な機能を提供するリアルタイムOS5と、OS5によって管理され実行される複数のタスク12～14と、OS5によって管理されるアイドルタスク13と、OS5がタスク10～13の実行制御を管理するために必要なデータを格納するためのタスク管理用データベース6とを含む。OS5は、OS1と異なり割込処理機能は持たない。

【0021】このシステムはさらに、OS1とOS2との間に設けられ、OS1とOS5との間で、たとえば特定の処理の実行の依頼などを通信するための通信手段14を含む。

【0022】図1に示されるデータ処理システムは以下

のように動作する。OS1は、図6に示されるリアルタイムOS60と同様に動作する。すなわち、OS1は、その管理するタスク7～9がシステムコールを発行すると、OS1に対する割込を禁止してから、このシステムコールの処理を行なう。このとき、前述のようにOS1が提供する機能は制限されたものである。そのため、システムコール処理時にデータベース2を操作する量は、提供する機能が複雑な場合と比べて少なくすることができる。したがってOS1がシステムコールを処理するために要する時間は短くなる。前述のようにシステムコール処理時には、割込が禁止される。したがってシステムコール処理時間が短くなることによりOS1に対する割込禁止時間も短くでき、割込に対する応答性が向上する。

【0023】図1に示されるシステムが従来のシステムと異なるのは、OS5を中心とするシステムがアイドルタスク4としてOS1の管理下で動作することである。OS1の管理下で動作するタスク7～9がなくなったときにOS5に実行が移り、OS5が管理するタスク10～13が実行される。タスク10～13がシステムコールを発行したときは、OS5がその処理を行なう。

【0024】OS5で行なわれるシステムコール処理のフローチャートを図3に示す。図3を参照して、システムコールが発生すると、ステップS31において実行中のタスクのコンテキストがタスク管理用データベース6にセーブされる。ステップS32において、OS5によってタスクのスケジューリングが行なわれる。このとき、タスク管理用データベース6のデータが操作される。続いてステップS33で、スケジューリングされた新タスクのコンテキストがタスク管理用データベース6からタスク実行用のハードウェアのレジスタ群にリストアされ、システムコール処理が終了する。

【0025】図3と図8とを対比してすぐわかるように、OS5によるシステムコール処理においては、システムコール実行中でも割込が禁止されない。これは、本発明の実施例に係るデータ処理システムにおいては、図1に示されるように、割込処理ルーチン3がOS1の管理下でのみ行なわれ、OS5においては割込処理を行なう必要がないためである。割込処理によってOS5のタスク管理用データベース6が操作されることはない。したがってOS5システムコール処理においては、割込禁止とする必要はない。OS5の機能をどれだけ複雑にしても、OS5のシステムコール処理実行中に割込禁止とする必要はないため、このデータ処理システム全体の割込に対する応答性を劣化させずにすむ。それに対してOS1においては割込処理を行なう必要がある。そのため、OS1は、制限された機能のみタスクに提供する。これによりシステムコール処理時にOS1がデータベース2を操作する量は少なくなり、OS1のシステムコール処理時間が短くなる。割込禁止時間も短くすることが

でき、割込に対する応答性を高めることができる。

【0026】さらに、浮動小数点計算など、大きなコンテキストを持つタスクはOS 5で、小さなコンテキストで実行可能なタスクはOS 1の管理下で、それぞれ実行させることができる。これにより、たとえば小さなコンテキストを有するタスクについてコンテキストのセーブ・リストアを行なう場合、従来よりも小さなコンテキストのセーブ・リストアのみでシステムコール処理を行なうことができる。大きなコンテキストのタスクについて

は、そのコンテキストのセーブ・リストアはOS 5で行なわれる。したがって、従来のように各タスクのコンテキストの大小にかかわらず最も大きなコンテキストに合わせたセーブ・リストアを行なうような無駄を省くことができ、OS 1の管理下のタスクからシステムコールを処理するための時間を短縮することができる。

【0027】一方、場合によっては、図1に示されるOS 1の管理下で実行されるタスクのいずれかから、システムコールを発行してOS 5に属するタスクを操作したい場合もあると考えられる。通信手段14はこれを可能とするために設けられたものである。OS 1に属するタスクが発行したシステムコールの処理は通信手段14を介してOS 1からOS 5に依頼される。そしてOS 5がこのシステムコールに対する処理を行なう。

【0028】この第1の実施例のデータ処理システムにおいては、OS 1のアイドルタスク4としてOS 5を割付けている。このようにすることにより、OS 1に属するタスク7～9には、OS 5に属するタスク10～12よりも高い優先度が自動的に与えられることになる。これは、以下のような理由による。OS 1は、高速にシステムコールを処理するためにその提供する機能を少なくし、また各タスクのコンテキストを軽くしている。つまり、高速に実行したいタスクはOS 1の管理下で実行させることが前提とされている。このような条件の下では、OS 1に属するタスクは、当然OS 5に属するタスクよりも高い優先度を有するべきである。

【0029】図4は、この発明の第2の実施例に係るデータ処理システムのソフトウェア構成図である。図4に示されるデータ処理システムが図1に示されるデータ処理システムと異なるのは、第2のOS 5を中心とするシステムが、第1のOS 1の、アイドルタスク以外のタスク4に割付けられることである。その他の点において、図4に示されるシステムは図1に示されるシステムと同様である。同一の構成要素には同一の参照番号および名称が与えられている。それらの機能も同一である。したがって、ここではそれらについての詳しい説明は繰り返されない。

【0030】図4に示されるシステムにおいては、第2のOS 5に属するタスクの優先度が自動的に第1のOS 1に属するタスクの優先度よりも低くなるということはない。この優先度はOS 1に割付けられる各タスクの優

先度と、OS 5に従属する各タスクに割当てられる優先度とから決まる。このようにすることによっても、OS 1による割込処理に対する高速応答が実現でき、またOS 1に従属するタスクとしてはコンテキストの小さなものを選択することにより、システムのオーバーヘッドが増加するおそれがなく、リアルタイムOSとして望ましい高速応答性を備えることができる。

【0031】図5には、本発明に係るデータ処理システムの第3の実施例のソフトウェア構成図が示されている。図5に示されるデータ処理システムが図1に示されるデータ処理システムと異なるのは、アイドルタスク4内のシステムのOS 5に従属するタスクのうちの1つが、第3のOS 51を含むことと、通信手段14がOS 1、5に加えてOS 51とも通信可能なこととである。

【0032】図5を参照して、アイドルタスク13は、OS 5の管理下で動作するOS 51と、OS 51の管理下で動作するタスク53～54と、OS 51がタスク53～54の実行制御を管理するために使用するタスク管理用データベース52とを含む。

【0033】タスク54はOS 51の管理下で動作するアイドルタスクである。OS 51を中心とするシステムとOS 5を中心とするシステムとの関係は、OS 5を中心とするシステムとOS 1を中心とするシステムとの関係と同様である。このシステムでは優先度はOS 1の従属タスク、OS 5の従属タスク、OS 51の従属タスクという順になる。各OS 1、5、51には、順にコンテキストの小、中、大のタスクが割当てられる。このようにすることにより、各タスクのコンテキストの大小に応じてシステムコール処理時のデータのセーブ・リストア量を変更することができる。そのため、システムコール処理時のシステムのオーバーヘッドが減少するとともに、OS 1のみで行なわれる割込処理に対する応答性も良好なものに保たれる。また、4つ以上のOSを使用し図5に示されるようなシステムを構成することもできる。

【0034】以上のようにこの発明に係るデータ処理システムにおいては、互いに独立したタスク管理用のデータベースを備えた複数のOSが使用される。割込処理はそのうちの1つのOSでのみ行なわれる。他のOSでは割込処理が行なわれない。そのため他のOSのシステムコール処理が複雑になっても、割込応答性を劣化させずにすむ。また、各タスクをそのコンテキストの大きさに応じて適切なOSの管理下で動作させることができる。したがって冗長なコンテキストのセーブ・リストアを省くことができ、システムのオーバーヘッドを取除くことができる。

【0035】

【発明の効果】以上のようにこの発明に係るデータ処理システムによれば、第1および第2の実行制御手段に、それぞれ独立の第1および第2の実行制御データ管理手

段を設けた。したがって各仕事単位を、その実行制御に必要なコンテキストの大きさに従って第1または第2の実行制御手段のうちの適切なものの管理下で動作させるように割振ることができる。冗長なコンテキストのセーブ・リストアを省くことが可能となり、システムの動作をより効率よくでき、その応答性も良好にすることができる。また、第1の実行制御手段と第2の実行制御手段とによって提供される機能を、その複雑さにおいて異なったものとすることができる。したがって第1の実行制御手段より提供される機能をより簡単なものとすることにより、第1の実行制御手段の管理下の第1の仕事単位についての処理を高速化できる。また、第2の実行制御手段により提供される機能をより複雑なものとするにより、より多くの機能を実行制御手段に要求するような第2の仕事単位も実行することが可能である。

【0036】その結果、システムの機能がより高度化しても、高速応答性が低下しないデータ処理システムを提供することができる。

【図面の簡単な説明】

【図1】図1は、本発明の第1の実施例に係るデータ処理システムのソフトウェア構成図である。

【図2】図2は、本発明の第1の実施例に係るデータ処理システムのハードウェア構成図である。

【図3】図3は、第1の実施例の、第2のOSで行なわ

*れるシステムコール処理のフローチャートである。

【図4】図4は、本発明の第2の実施例に係るデータ処理システムのソフトウェア構成図である。

【図5】図5は、本発明の第3の実施例に係るデータ処理システムのソフトウェア構成図である。

【図6】図6は、従来のリアルタイムOSを用いたデータ処理システムのソフトウェア構成図である。

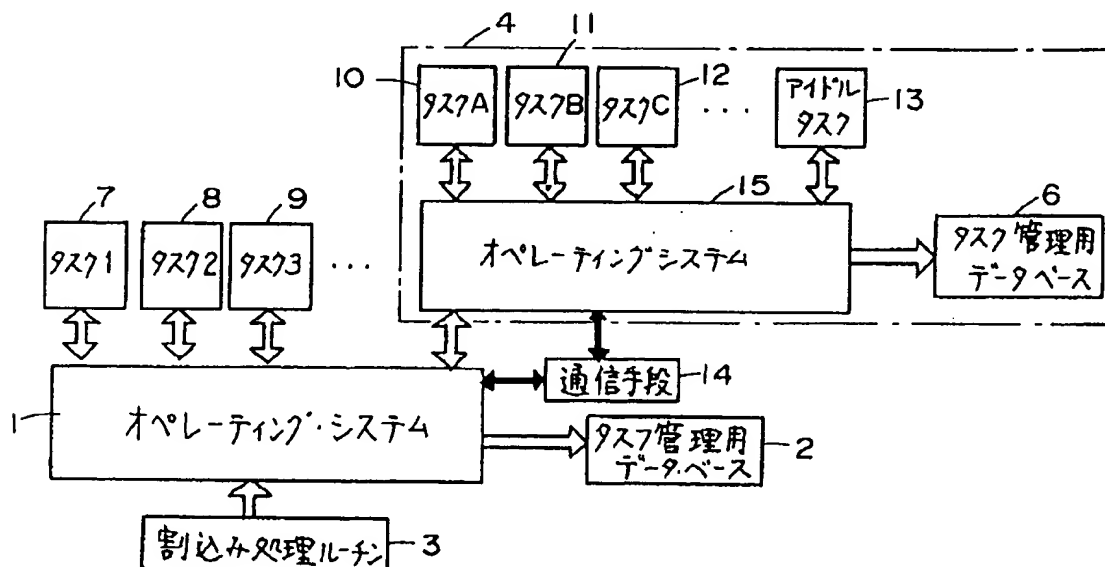
【図7】図7は、タスクのディスパッチの様子を示す模式図である。

【図8】図8は、従来のリアルタイムOSのシステムコール処理のフローチャートである。

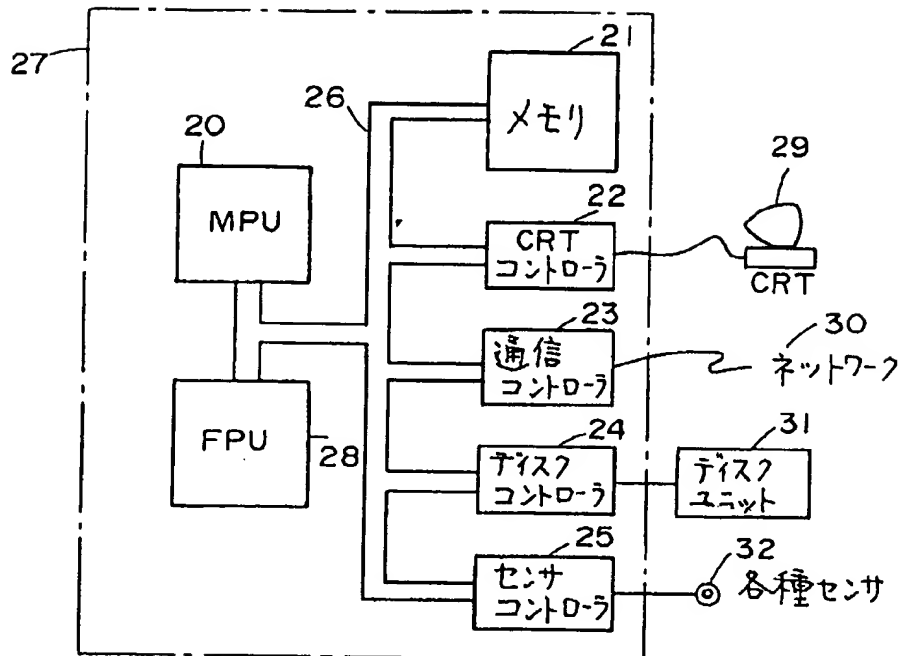
【符号の説明】

- 1 リアルタイムOS
- 2 OS1が使用するデータベース
- 3 割込処理ルーチン
- 5 割込処理に関与しないOS
- 6 OS5が使用するデータベース
- 7～9 OS1の管理下で動作するタスク群
- 10～12 OS5の管理下で動作するタスク群
- 13 OS5のアイドルタスク
- 14 OS1とOS5の間の通信手段
- 20 マイクロプロセッサユニット
- 21 メモリ
- 26 内部バス

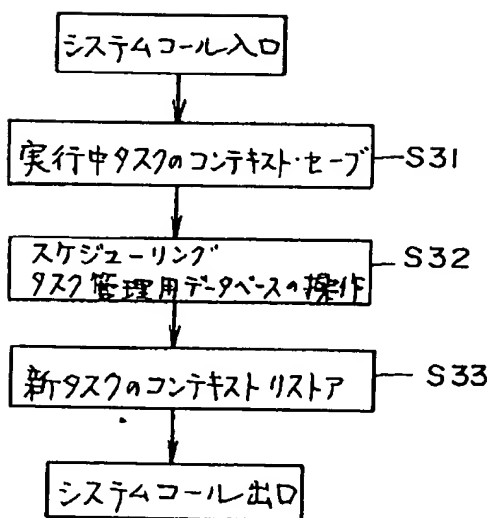
【図1】



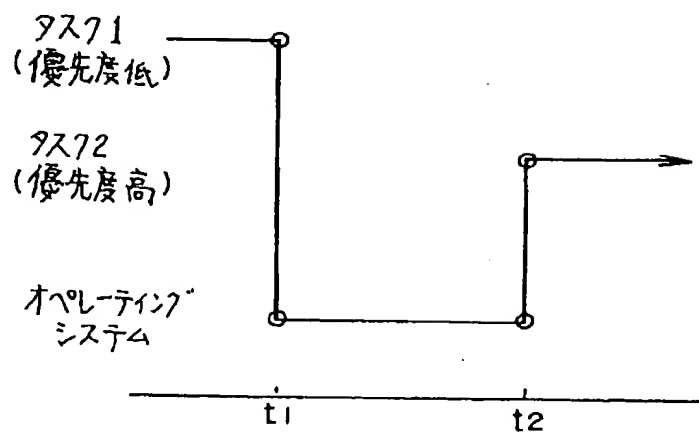
【図2】



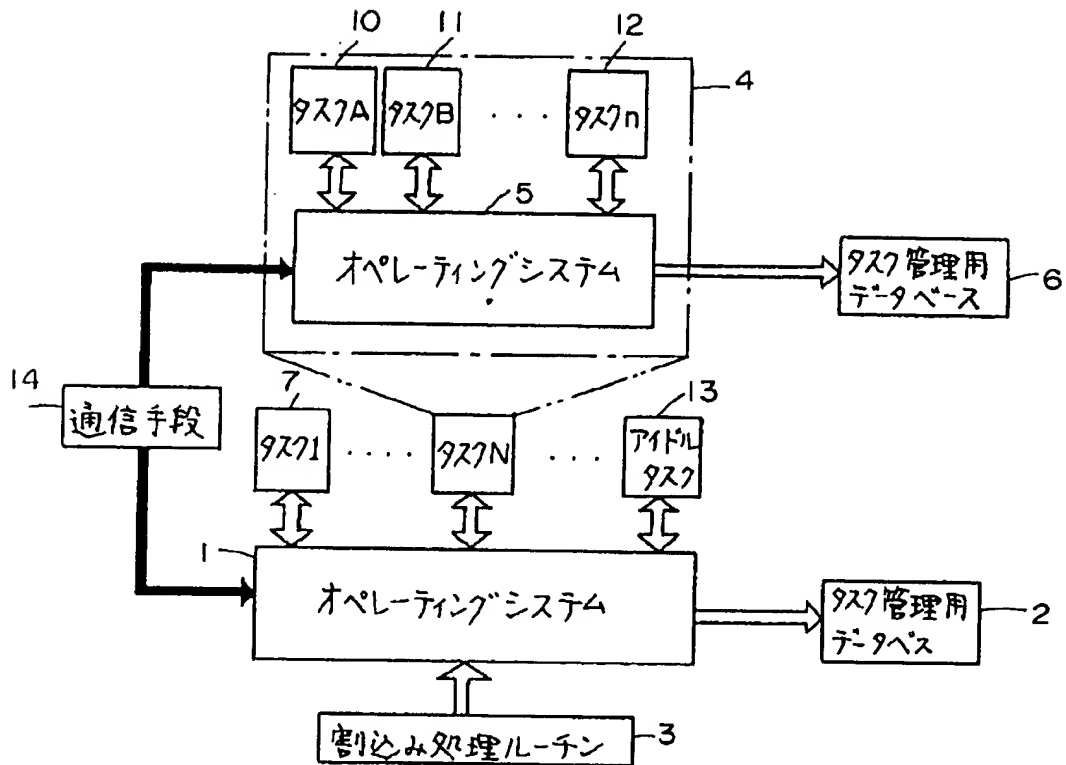
【図3】



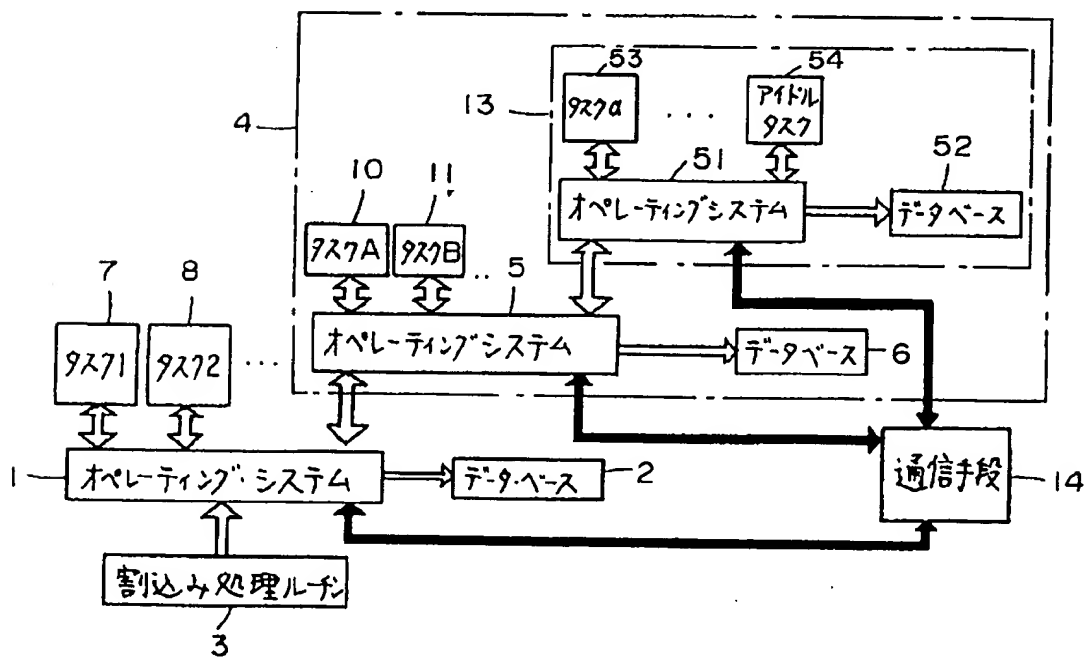
【図7】



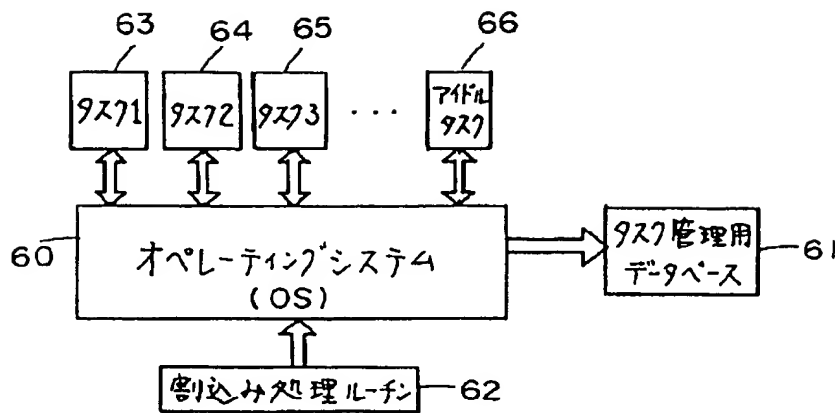
【図4】



【図5】



【図 6】



【図 8】

